The match will be a unique match for a telephone extension, but may not be a unique match for a name. If the match is not unique, system **101** returns the subscriber names of all matching records, and a prompt to select one of them, via API **204**, **208** to daemon process **206**, at step **432**. Upon receiving the matching names and prompt, at step **434**, daemon process **206** forwards them along with a return Internet/intranet address of daemon process **206** via Internet/intranet **102** to the return address of terminal **103**, at step **436**. Browser **104** receives the matching names and prompt and displays them to the user, at step **438**. When the user makes a selection of one of the names, browser **104** returns an indication of the selection to daemon process **206** via Internet/intranet **102**, at step **440**. Upon receiving the selection, at step **442**, daemon process **206** forwards it via API **204**, **208** to the return address of daemon process **206** at system **101**, at step **444**. Upon receipt of the selection, at step **446**, or if a unique match had been found at step **430**, conventional processes of system **101** return the subscriber ID of the selected matching subscriber to daemon process **206** via API **204**, **208**, at step **448**.

Turning now to FIG. **5**, daemon process **206** receives the subscriber ID, at step **500**, and in response requests the subscriber record for that subscriber ID from system **101**, at step **502**. Conventional processes of system **101** respond to receipt of the request, at step **504**, by retrieving and returning the requested record, at step **506**. Upon receipt of the record, at step **508**, daemon process **206** logs out of system **101**, at step **510**, and then proceeds to dynamically generate a Web home page for the subscriber from the received record, at step **512**. Illustratively, daemon process **206** creates the subscriber's home page by populating fields **701** of a generic Web home page template **700** (i.e., a form) with the subscriber's information, including the subscriber ID, from the received record. An example of such a generic Web home page template **700** is shown in FIG. **7**. Alternatively, subscribers of messaging system **101** may be allowed to create and store, linked to their records in subscriber directory **203**, their own custom Web home pages that are functionally equivalent to a populated page template **700**, in which case daemon process **206** merely retrieves the identified subscriber's custom home page at steps **502–512**. Daemon process **206** then uses the TCP/IP protocol to send the subscriber's home page over Internet/intranet **102** to the return address of terminal **103** which had sent the original request at step **400**, at step **514**, and then discards the subscriber's home page, at step **516**. If and when it receives another request for that subscriber, daemon process **206** will again retrieve the subscriber's record and create the home page anew, thus ensuring that any changes that might have been made to the subscriber's record (e.g., a new active personal greeting) will be reflected in the newly-requested home page.

The subscriber's home page that was sent by system **101** at step **514** is received by browser **104** and is displayed to the user on the display of terminal **103**, at step **520**. If any of the fields of the home page (e.g., "Today's greeting") include audio information, displaying of the home page on terminal **103** may include an HTML anchor including the "~" meta-character for playback of the audio information to the user. When the user selects the anchor, the audio information is provided by the server via the "~" meta-character processing as described. To send a message back to the subscriber, the user fills out the message fields **702** of the displayed home page, at step **522**. Typically, the fields filled out by the user are at least a header field **703** that indicates who the message is from, the subject of the message, the user's return address etc., and a message-body text field **704**, as shown in FIG. **7**. When the user is done entering the message, he or she invokes a message-send function, illustratively by pointing to and clicking on a "SEND MESSAGE" virtual button **705** of the displayed home page, at step **524**. The "SEND MESSAGE" button **705** is coded with the URL of a CGI script **207** which invokes the send-it process **205** of system **101**. In response, browser **104** sends data files of the message fields **702** filled out by the user, the subscriber ID of the intended message recipient, and the return address of terminal **103**, as a message to the URL of send-it process **205** at system **101** via Internet/intranet **102**, at step **526**.

Turning to FIG. **6**, HTTP server daemon process **206** receives the message files at system **101**, at step **600**, and in response to the URL of send-it process **205** invokes send-it process **205** of system **101**, at step **602**. Upon invocation of send-it process **205**, at step **604**, daemon process **206** forwards the message files and the subscriber ID of the subscriber who is to receive the message, at step **606**. Upon receipt of the message files and subscriber ID from daemon process **206**, at step **608**, send-it process **205** uses the received information to compose a message in the conventional format of system **101**, at step **610**. Send-it process **205** then logs into system **101**, at step **612**, and when system **101** accepts the login, at step **614**, send-it process **205** delivers the message to system **101** addressed to the recipient subscriber's ID, at step **616**. System **101** accepts the message, at step **618**, and delivers it to mailbox **200** of the recipient subscriber, at step **620**. System **101** then returns an acknowledgment of successful message delivery to send-it process **205**, at step **622**. In response to receipt of the acknowledgment, at step **624**, send-it process **205** logs off of system **101**, at step **626**, and system **101** accepts and completes the log off, at step **628**. Send-it process **205** then deletes the message files that it had received at step **608**, at step **628**, creates an acknowledgment message indicative of successful delivery of the message and addressed to the return address of terminal **103**, at step **630**, and forwards the acknowledgment message to daemon process **206**, at step **632**. Daemon process **206** responds to receipt of the acknowledgment message, at step **634**, by sending a Web page containing the message via the TCP/IP protocol over Internet/intranet to terminal **103**, at step **636**. Terminal **103** receives the Web page and browser **104** displays it to the user/message sender on the display of terminal **103**, at step **638**. The acknowledgment message created at step **630** may be a negative acknowledgment, e.g., "This mailbox is full. Message unable to be delivered." The message sender is thus given substantially immediate confirmation of the delivery or non-delivery of his or her message.

Of course, various changes and modifications to the illustrative embodiment described above will be apparent to those skilled in the art. For example, protocols other than the Web may be employed to effect the same type of service. Of specific interest are the FTP protocol, which allows the simple transfer of files between computers, Telnet, which supports terminal emulation and login to host capabilities, and Chat, which is a very simple split-screen two-way typing application allowing two people to type at each other simultaneously. This provides the requisite infrastructure for text and binary file message delivery. An illustrative implementation for each of these protocols is outlined by the following steps:

For FTP:

1. Set up an FTP server with a directory for each extension having a corresponding enabled mailbox. Symbolically link a directory of the extension's corresponding username to the extension directory, to enable addressing of the extension